**Hacker Guardian**

**（HG）**

**Design Specification**

**Stanford Zhang**

[**zhangstanford@gmail.com**](mailto:zhangstanford@gmail.com)

[**stanfordzhang@qq.com**](mailto:stanfordzhang@qq.com)

**Stanford Studio**

目 录

[1. Abstract 1](#_Toc352769271)

[2. References 1](#_Toc352769272)

[3. Developing Log 1](#_Toc352769273)

[4. 概要设计 2](#_Toc352769274)

[4.1. Process List(Ring 0) 3](#_Toc352769275)

[4.2. Process List(Ring 3) 3](#_Toc352769276)

[4.3. Driver List 3](#_Toc352769277)

[4.4. Kernel Info 4](#_Toc352769278)

[4.5. Kernel Hook 4](#_Toc352769279)

[4.6. User Mode Hook 4](#_Toc352769280)

[5. Hacker Guardian详细设计 5](#_Toc352769281)

[5.1. HG启动方式和系统版本检查 5](#_Toc352769282)

[5.2. Ring3和Ring0通信 6](#_Toc352769283)

[6. Windows Ring0技术剖析 6](#_Toc352769284)

[6.1. Develop HG GUI by using WTL 6](#_Toc352769285)

[6.2. Develop hgdll.dll by exporting one interface 6](#_Toc352769286)

[6.3. Reload ntoskrnl.exe in ring0 6](#_Toc352769287)

[6.4. Reload hal.dll in ring0 6](#_Toc352769288)

[6.5. Reload win32k.sys in ring0 6](#_Toc352769289)

[6.6. Ring0枚举进程（1） - ZwQuerySystemInformation 6](#_Toc352769290)

[6.7. Ring0枚举进程（2） - PspCidTable 6](#_Toc352769291)

[6.8. ZwQuerySystemInformation完全解析 6](#_Toc352769292)

# Abstract

Hacker Guardian是一款主要用于Anti-Rootkit/Virus的辅助工具软件。本程序所面向的用户是专业的Windows操作系统安全领域软件开发、病毒/木马分析等人员。

本程序是作者在学习Windows Kernel Development时的一个附属作品。开发过程中所涉及的大量技术都是来自互联网热爱分享的同仁们，在这里我仅代表我自己对这些乐于“思考 + 分享”知识的同仁们表示崇高的敬意。在文中和代码中我将会尽量把我引用代码、技术知识的出处标注上，但由于参考的信息太多，难免有所遗漏，在这里我对原作者表示深深的歉意，如果有可能作者可以联系我，负责任的将出处补上。

出于“思考 + 分享”的精神，我将会把整个工程的源代码、设计文档等资源上传到GitHub上开源分享，同时喜欢更多的“牛人”、有识之士共同交流，促进发展。

# References

1. A-Protect
2. AntiSpy

# Developing Log

|  |  |  |
| --- | --- | --- |
| Date | Author | Mission |
| 2013-04-03 | Stanford | 创建设计说明书，撰写其中的7.1、7.2和8三个章节。 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 概要设计

Hacker Guardian包含如下功能：

HackTitan is an anti-virus and root kit tool to professional.

“dll” directory is used to place dll files;

“driver” directory is used to place driver(sys) files;

“include” directory is used to place shared include(.h) files.

“plugin” directory is used to place plugin module files(dll).

“tool” directory is used to place utility tools(dll);

Support Windows Version:

Windows XP 32-bit

Windows 7 32-bit

Windows 7 64-bit

Windows 8 32-bit

Windows 8 64-bit

主程序Crash Dump文件自动发送到一个email邮箱。

获取进程列表的方式分为Ring0和Ring3.

对于每一个进程都可以查看PE信息。

工具：带有对话框的DLL接口规格。

插件：不带有对话框的DLL接口规格。

界面显示方案

分两个窗口：

左侧为树形菜单；右侧为菜单中的内容。

功能分类：

## Process List(Ring 0)

**1.1 Un-document API**

**1.2 Inject DLL/Code to a process**

**1.3 Create Dump**

## Process List(Ring 3)

**2.1 Module List**

**2.2 Thread List**

**2.3 Handle List**

**2.4 Window List**

**2.5 Memory List**

**2.6 Hotkey List**

**2.7 Timer**

**2.8 Terminate Process**

**2.9 Force Terminate Process**

**2.10 Suspend Process**

**2.11 Resume Process**

**2.12 Properties**

**2.13 Dump**

**2.14 PE Info**

**2.15 Inject**

## Driver List

**3.1 Copy Module Memory**

**3.2 Delete Driver (File)**

**3.3 Delete Driver (File & Reg)**

**3.4 Unload Driver**

**3.5 Properties**

## Kernel Info

**4.1 System Callback**

**4.2 Filter Driver**

**4.3 DPC Timer**

**4.4 Work Queue Thread**

**4.5 HAL**

**4.6 File System**

**4.7 System Debug**

**4.8 Object Hijack**

**4.9 Directive IO**

**4.10 GDT**

## Kernel Hook

**5.1 SSDT**

**5.2 Shadow SSDT**

**5.3 FSD**

**5.4 Keyboard**

**5.5 Mouse**

**5.6 Disk**

**5.7 Atapi**

**5.8 Acpi**

**5.9 Scsi**

**5.10 Kernel Hook**

**5.11 Object Hook**

**5.12 System Interrupt**

## User Mode Hook

**6.1 Message Hook**

**6.2 Process Hook**

**6.3 Kernel Callback Table**

# Hacker Guardian详细设计

## HG启动方式和系统版本检查

因为程序分为两个版本：32Bit和64Bit，所以不同位数位的Windows系统启动不同的程序。

在这里约定启动程序的名字为HackerGuardian.exe，32Bit的程序为hg32.exe；64Bit位的程序为hg64.exe。

* HackerGuardian.exe

1. 使用Masm32汇编编写的32Bit程序，因为Windows 64Bit系统是兼容32Bit程序的，由这个EXE来判断启动hg32.exe还是hg64.exe。
2. 同时该EXE还将进行系统版本检查，因为本程序涉及许多Undocumented API和许多未公开的技术，所以对于每一个支持的操作系统都有严格的要求。本程序只支持指定的测试通过的如下几个Windows版本系统：

* Windows 2000；
* Windows XP;
* Windows Server 2003;
* Windows Server 2003 SP1;
* Windows Server 2003 SP2;
* Windows Vista;
* Windows Vista SP1;
* Windows Vista SP2;
* Windows 7;
* Windows 8。
* Hg32.exe

Hg32.exe所涉及到的所有模块都是32Bit环境下编译的。

* Hg64.exe

Hg64.exe所涉及到的所有模块都是64Bit环境下编译的。

## Ring3和Ring0通信

Ring3和Ring0通信过程中，如果需要动态分配内存，则Ring3分配好内存再映射到Ring0。

但在实际开发过程中很多时候并不知道Ring0 Response的数据是多少，从而Ring3就不知道需要分配多少内存。在这里参考ZwQuerySystemInformation的方式设计一种方案，Hacker Guardian中统一使用这种方法：

1. Ring3操作中将分配的内存指针赋为NULL，同时给一个Optional字段用于存放返回值；
2. Ring0判断已分配的内存指针是不是NULL，如果是NULL则计算需要分配多少内存，将值存放在Optional字段中返回给Ring3，然后Ring3再分配相应的内存，并将地址传递给Ring0。
3. Ring0中获取某个数据有多个方法，多个方法所得结果可能会有些差异，此时对于1，2两步的操作不太方便。

# Windows 系统架构

# 构建开发环境

# Windows 驱动开发觉问题详解

# Windows Ring3开发

## WTL

## PE

# Windows Ring0开发

## Develop HG GUI by using WTL

## Develop hgdll.dll by exporting one interface

## Reload ntoskrnl.exe in ring0

## Reload hal.dll in ring0

## Reload win32k.sys in ring0

## Ring0枚举进程（1） - ZwQuerySystemInformation

## Ring0枚举进程（2） - PspCidTable

## 调用ZwQuerySystemInformation枚举进程

## EPROCESS剖析

## Idle Process和System Process剖析

## 查找Idle Process的EPROCESS

## 查找System Process的EPROCESS

## Windows 8 32/64-Bit中关于PspCidTable的变更

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Author(s) : Stanford Zhang

\* E-Mail : zhangstanford@gmail.com

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

最近学习通过 PspCidTable 遍历进程列表，分享一点遍历 PspCidTable 在Windows 8 32/64-Bit 中的变化。网上关于 PspCidTable 的文章多如牛毛，在这里就不再赘述，可参考百度和 Google，在这里推荐 sysdog 的《PspCidTable完全解读》。

本文中只详细查看了 Windows XP 32-Bit、Windows 7 32/64-Bit、Windows 8 32/64-Bit 五个版本的Windows相关内核信息，如果其他系统略有差异，请具实际情况而论。PspCidTable 在 Windows XP 32-Bit 和 Window 7 32/64-Bit 中两个结构 \_HANDLE\_TABLE 和 \_HANDLE\_TABLE\_ENTRY 没有变化，只是在 64-Bit 系统中一些指针和字段偏移变成64-Bit，其他都是一样的（下文中选中Windows XP 32-Bit 为被比较对象），其遍历方法请参考网上相关文章。

1. **PspCidTable变化之 Windows 8 32/64-Bit - \_HANDLE\_TABLE**

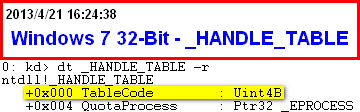


图1-1 Windows 7 32-Bit \_HANDLE\_TABLE结构

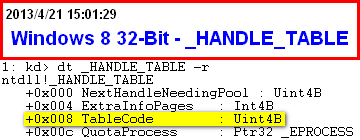


图1-2 Windows 8 32-Bit \_HANDLE\_TABLE结构

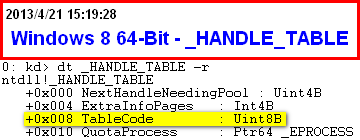


图1-3 Windows 8 64-Bit \_HANDLE\_TABLE结构

如以上3图所示，Windows 8中的 \_HANDLE\_TABLE 与其他系统中的 TableCode 字段偏移不一样，所以在查找地址的时候要加上偏移（0x008），且 Windows 8 32-Bi t和 64-Bit TableCode 字段的偏移是一样。TableCode 字段的解释与其他系统中一样的，低 3 Bits 表示是几层表结构，低3位清0之后的值就是层表地址。

1. **PspCidTable变化之 Windows 8 64-Bit - \_HANDLE\_TABLE\_ENTRY**

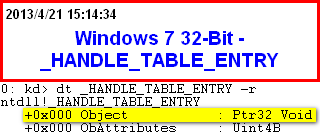


图2-1 Windows 7 32-Bit \_HANDLE\_TABLE\_ENTRY结构

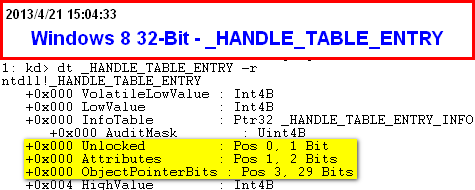


图2-2 Windows 8 32-Bit \_HANDLE\_TABLE\_ENTRY结构

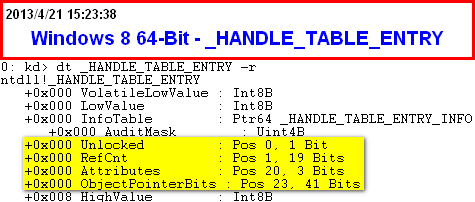


图2-3 Windows 8 64-Bit \_HANDLE\_TABLE\_ENTRY结构

如以上图所示，Windows 7 32-Bit中的\_HANDLE\_TABLE\_ENTRY和Windows 8 32-Bit/64-Bit都不一样。但在实验过程中发现，Window 7 32-Bit 与Windows 8 32-Bit中偏移0x000处的4个字节的解释是一样的，只不过Windows 8 32-Bit中的结构用位域将信息详细表达了出来。从Windows 8 32-Bit中可看出低3 Bits另有用处，在解释的时候和Windows 7 32-Bit是一样的，将低3 Bits清零和最高Bit置1。计算公式如下：

Object Address = (Object/ObjectPointerBits & 0xFFFFFFF8) | 0x80000000

注：这里的ObjectPointerBits是Windows 8 32-Bit中的字段。

如图2-3中所示，Windows 8 64-Bit 中Unlocked字段没变，但多了一个RefCnt（Windows 8 32-Bit中RefCnt字段是在偏移为0x004的4字节中，从第26 Bit开始的6 Bits）字段且占从第1 Bit开始的19 Bits，Attributes字段名称没变但从原来占2 Bits变成占3 Bits（从第20 Bit开始），ObjectPointerBits占位从第23 Bit的41 Bits。同时，1 + 19 + 3 + 41 = 64，正是64-Bit系统中地址所占位数。

这里变化最大的是ObjectPointerBits字段，Windows 64-Bit系统中地址占64 Bits中的44 Bits，所在这里猜测将\_HANDLE\_TABLE\_ENTRY的前8个字节右移19位，这样结构就和Windows 8 32-Bit系统中的结构是一样的了，所以得以下计算公式：

Object Address = ((ObjectPointerBits >> 19) & (~0xF)) | 0xFFFFF00000000000

经过实验验证，在这之后对这个地址进行处理就可以得到相应的进程之类的信息。

注：由于实验资源有限，只在两台Windows 8 x64系统上经过测试无误，如果在其他系统中有误，请告知，谢谢。